图文 77 如果优惠券系统的数据库宕机，如何用死信队列解决这种异常场景？

详情 评论

如果优惠券系统的数据库宕机，如何用死信队列解决这种异常场景？

继《从零开始带你成为JVM实战高手》后，救火队长携新作再度出山，重磅推荐：

（点击下方蓝字试听）

《从零开始带你成为MySQL实战优化高手》

目录

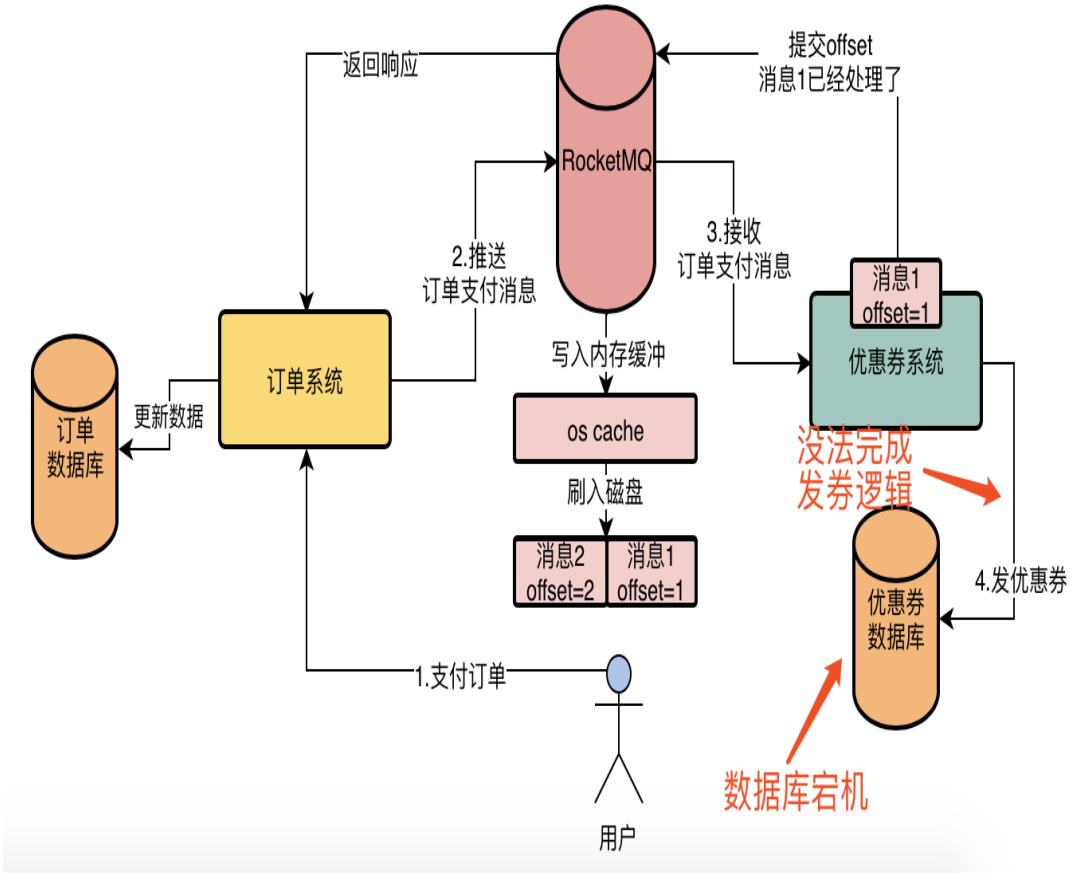
1、如果优惠券系统的数据库宕机，会怎么样？

之前我们已经分析和解决了MQ实践使用过程中可能存在的消息丢失问题和消息重复问题，现在假设我们可以基本确保MQ的消息不丢失，同时不会对消息进行重复处理，在正常流程下，基本没什么问题了。

那么接着我们来看下一个问题，假设我们的MQ使用都没问题，但是如果我们的优惠券系统的数据库宕机了呢？

因为我们一直都是假设了一个场景，就是订单支付成功之后会推送消息到MQ，然后优惠券系统、红包系统会从MQ里获取消息去执行后续的处理，比如发红包或者发优惠券。

那么如果这个时候，优惠券系统的数据库宕机了，就必然会导致我们从MQ里获取到消息之后是没办法进行处理的，我们看下图。



所以针对这样的一个坑爹的异常场景我们应该怎么处理？优惠券系统应该怎么对消息进行重试？重试多少次才行？万一反复重试都没法成功，这个时候消息应该放哪儿去？直接给扔了吗？

我们今天就对这个实际的生产场景进行分析。

2、数据库宕机的时候，你还可以返回CONSUME_SUCCESS吗？

先让我们回顾一下你的优惠券系统使用RocketMQ的Consumer是如何从MQ中获取到消息的，我们看下面的代码片段

在下面的代码片段中，清晰可以看到，我们注册了一个监听器回调函数，当Consumer获取到消息之后，就会交给我们的函数来处理。

```
consumer.registerMessageListener(
        new MessageListenerConcurrently() {
                @Override
                public ConsumeConcurrentlyStatus consumeMessage(
                        List<MessageExt> msgs,
                        ConsumeConcurrentlyContext context) {
                    // 在这里对获取到的msgs订单消息进行处理
                    // 比如增加积分、发送优惠券、通知发货，等等
                    return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
                }
        }
);
```

而且我们之前还对这个方法进行了分析，我们可以在这个回调函数中对消息进行处理，比如发红包、发优惠券之类的，处理完成之后，就可以返回一个状态告诉RocketMQ Consumer这批消息的处理结果。

比如，如果返回的是CONSUME_SUCCESS，那么Consumer就知道这批消息处理完成了，就会对提交这批消息的Offset到broker去，然后下次就会继续从broker获取下一批消息来处理了。

但是如果此时我们在上面的回调函数中，对一批消息发优惠券的时候，因为数据库宕机了，导致优惠券发放逻辑无法完成，此时我们还能返回CONSUME_SUCCESS状态吗?

如果你返回的话，下一次就会处理下一批消息，但是这批消息其实没处理成功，此时必然导致这批消息就丢失了。

肯定会导致有一批用户没法收到优惠券的!

3、如果对消息的处理有异常，可以返回RECONSUME_LATER状态

所以实际上如果我们因为数据库宕机等问题，对这批消息的处理是异常的，此时没法处理这批消息，我们就应该返回一个RECONSUME_LATER状态

他的意思是，我现在没法完成这批消息的处理，麻烦你稍后过段时间再次给我这批消息让我重新试一下!

所以我们看下面的代码，应该改成如下的方式:

```
consumer.registerMessageListener(
       new MessageListenerConcurrently() {
              @Override
              public ConsumeConcurrentlyStatus consumeMessage(
                     List<MessageExt> msgs,
                     ConsumeConcurrentlyContext context) {
                 try {
                     // 在这里对获取到的msgs订单消息进行处理
                     // 比如增加积分、发送优惠券、通知发货，等等
                     return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
                 } catch(Exception e) {
                     // 如果因为数据库宕机等问题，对消息处理失败了
                     // 此时返回一个稍后重试消费的状态
                     return ConsumeConcurrentlyStatus.RECONSUME_LATER;
                 }
              }
       }
);
```

大家可以在上面的代码中看到，我们已经做出了相应的修改，如果消息处理失败了，就返回RECONSUME_LATER状态，让RocketMQ稍后再重新把这批消息给我，让我重试对这批消息进行处理!

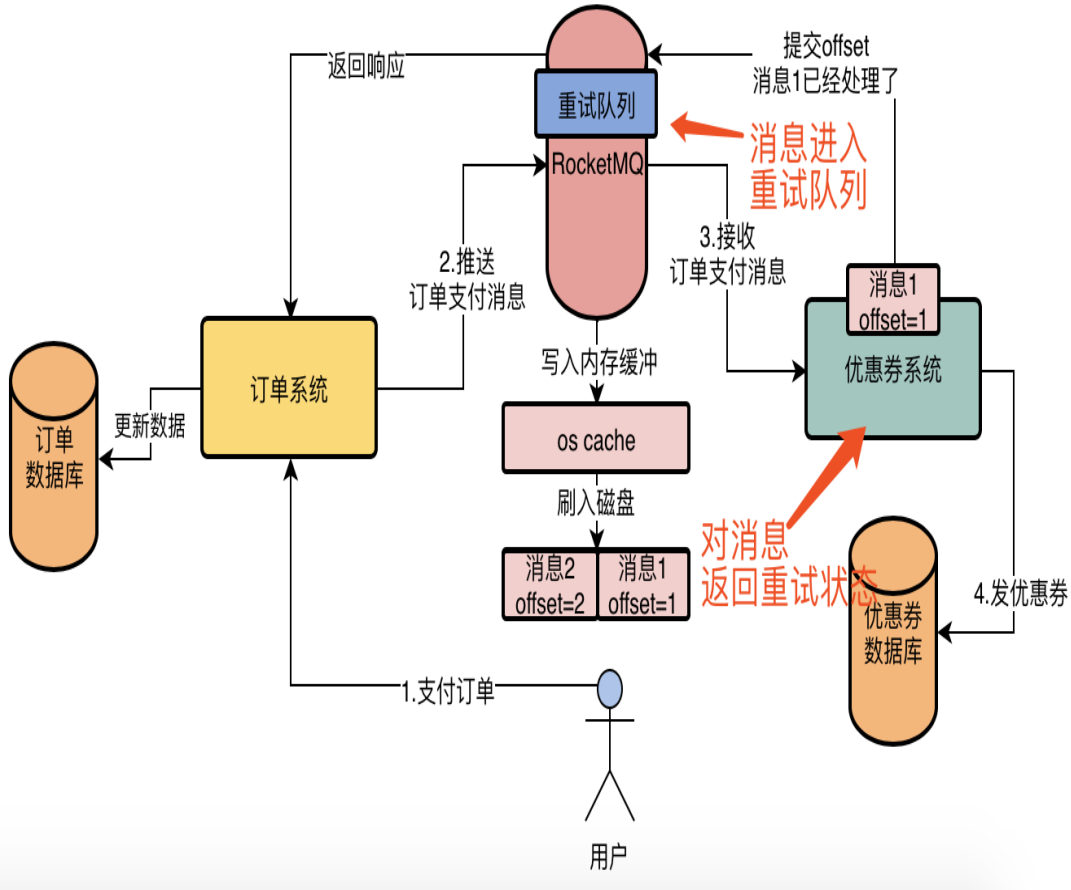
4、RocketMQ是如何让你进行消费重试的?

那么RocketMQ在收到你返回的RECONSUME_LATER状态之后，是如何让你进行消费重试的呢?

简单来说，RocketMQ会有一个针对你这个ConsumerGroup的重试队列。如果遗忘了ConsumerGroup消费组概念的朋友可以再回过头去复习一下。

如果你返回了RECONSUME_LATER状态，他会把你这批消息放到你这个消费组的重试队列中去

比如你的消费组的名称是“VoucherConsumerGroup”，意思是优惠券系统的消费组，那么他会有一个“%RETRY%VoucherConsumerGroup”这个名字的重试队列，我们看下图的示意。



然后过一段时间之后，重试队列中的消息会再次给我们，让我们进行处理。如果再次失败，又返回了RECONSUME_LATER，那么会再过一段时间让我们来进行处理，默认最多是重试16次！每次重试之间的间隔时间是不一样的，这个间隔时间可以如下进行配置：

messageDelayLevel=1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h

上面这段配置的意思是，第一次重试是1秒后，第二次重试是5秒后，第三次重试是10秒后，第四次重试是30秒后，第五次重试是1分钟后，以此类推，最多重试16次！

5、如果连续重试16次还是无法处理消息，然后怎么办?

那么如果在16次重试范围内消息处理成功了，自然就没问题了，但是如果你对一批消息重试了16次还是无法成功处理呢?

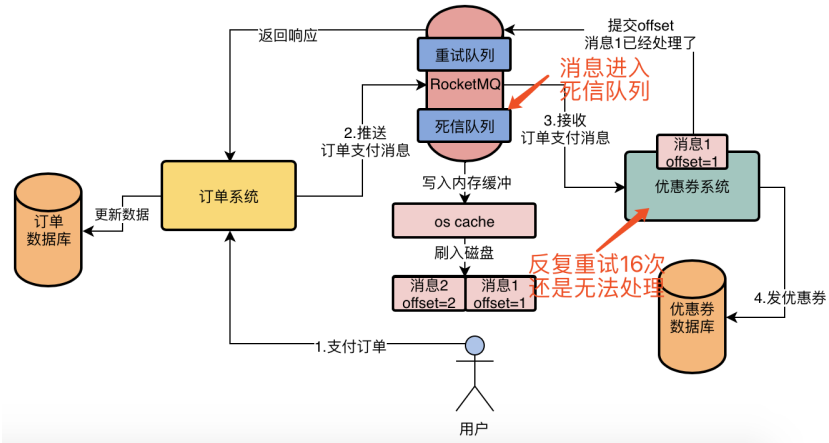
这个时候就需要另外一个队列了，叫做**死信队列**，所谓的死信队列，顾名思义，就是死掉的消息就放这个队列里。

那么什么叫死掉的消息呢?

其实就是一批消息交给你处理，你重试了16次还一直没处理成功，就不要继续重试这批消息了，你就认为他们死掉了就可以了。然后这批消息会自动进入死信队列。

死信队列的名字是“%DLQ%VoucherConsumerGroup”，我们其实在RocketMQ的管理后台上都是可以看到的。

如下图所示



那么对死信队列中的消息我们怎么处理?

其实这个就看你的使用场景了，比如我们可以专门开一个后台线程，就是订阅“%DLQ%VoucherConsumerGroup”这个死信队列，对死信队列中的消息，还是一直不停的重试。

6、消息处理失败场景下的方案总结

这一次我们就搞清楚了另外一个生产环境下的问题，就是消费者底层的一些依赖可能有故障了，比如数据库宕机，缓存宕机之类的，此时你就没办法完成消息的处理了，那么可以通过一些返回状态去让消息进入RocketMQ自带的重试队列，同时如果反复重试还是不行，可以让消息进入RocketMQ自带的死信队列，后续针对死信队列中的消息进行单独的处理就可以了。

End

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑

如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：《付费用户如何加群》（**购买后可见**）